1. ОРГАНИЗАЦИЯ ПОДПРОГРАММ. ПРОЦЕДУРЫ И ФУНКЦИИ

Мы уже видели, что Паскаль-программа состоит из последовательности операторов (простых и структурных). Есть в языке также понятие функции. Система TurboPascal и другие системы программирования на базе языка Паскаль имеют целый набор встроенных машинных функций. Однако Паскаль предоставляет возможность создавать новые операторы и функции на основе стандартных. Такие процедуры и функции принято называть п о л ь з о в а т е л ь с к и м и.

В Паскале два типа подпрограмм – процедуры и функции. Создание процедур и функций является дальнейшим развитием идеи группирования, когда несколько операторов объединяются в программный блок, который имеет свое имя и к которому можно обратиться по этому имени с помощью соответствующих средств вызова. Процедура вызывается с помощью оператора вызова процедуры с одноименным именем. Функция пользователя также имеет свое имя, и ее активизация происходит при вычислении выражения, содержащего имя функции. Возвращаемое функцией значение подставляется в это выражение.

1.1. Процедуры и их типизация

Итак, процедура – это часть программы (подпрограмма), имеющая имя и предназначенная для решения некоторой частной задачи (подзадачи). Процедуры делятся по способам описания и обращения к ним.

Процедура встроенная (*машинная*) – это процедура, описание которой считается известной транслятору, в связи с чем ее можно использовать в программе, зная только ее имя.

Процедура пользователя – процедура, которую создает (описывает) программист на основе имеющихся операторов и встроенных процедур и функций данного языка по определенным правилам данного языка.

Процедура без параметров – процедура, при обращении к которой не требуется задания начальных установок, значений и после выполнения которой в основную программу не передаются результаты работы данной процедуры.

Процедура с параметрами-значениями – процедура, при обращении к которой требуются только начальные значения. На выходе данные не передаются в основную программу.

Процедура с параметрами-переменными – процедура, не требующая начальных значений, однако передающая в основную программу результаты своей работы (передает значения некоторых переменных).

Комбинированная процедура – процедура, имеющая параметры-переменные и параметры-значения, т. е. входные и выходные данные.

1.1.1. Встроенные процедуры

Встроенные процедуры являются составной частью системы программирования. Среди них есть стандартные процедуры, которыми можно пользоваться в любом месте программы без какого-либо предварительного объявления. Сюда относятся уже ранее упомянутые процедуры ввода/вывода, управления работой программы, динамического распределения памяти, строковые процедуры и пр. Полный перечень встроенных процедур можно найти в справочнике для языка.

Помимо стандартных процедур в Паскале есть также стандартные модули, представленные в виде TPU – файлов, каждый из которых содержит в себе целый набор процедур и функций. Для того чтобы использовать процедуры из модулей, необходимо вызвать нужный модуль в разделе USES. Система TurboPascal имеет модули PRINTER, DOS, CRT, GRAPH и др.

CRT позволяет использовать все возможности дисплея и клавиатуры, вномерамая управление режимом работы экрана, расширенные коды клавиатуры, цвет, окна и звуковые сигналы.

DOS поддерживает различные функции ДОС, вномерамая установку и получение текущего значения даты и времени, поиск по каталогам файлов и выполнение программ.

PRINTER позволяет легко организовать доступ к устройству печати.

GRAPH – мощный графический пакет с набором процедур и функций обработки графических объектов (точек, отрезков, прямоугольников, окружностей и пр.).

Рассмотрим несколько примеров встроенных процедур:

- CLRSCR – процедура очистки экрана. Результатом работы является стирание всей информации с экрана. Данная процедура является примером процедур без параметров.
- GOTOXY (A, B) – процедура позиционирования курсора на экране дисплея в точку с координатами (A, B). A и B являются входными данными, следовательно, это пример процедуры с параметрами-значениями.

1.1.2. Процедуры пользователя

При работе с процедурами пользователя необходимо уметь производить два вида деятельности: описание процедуры и обращение к ней в основной программе. Вызов процедуры пользователя осуществляется так же, как и вызов встроенной процедуры, – с помощью оператора вызова процедуры, имя которого совпадает с именем процедуры, с ука-

занием списка параметров, если таковые имеются. Описание же процедуры включает в себя разработку подпрограммы и правильное оформление ее заголовка. Остановимся на нем более подробно.

В основной программе все процедуры (а также и функции) пользователя должны быть объявлены. Объявление процедур и функций осуществляется после объявления переменных и перед первым словом BEGIN программы.

Процедура, как видно из ее определения, оформляется так же, как и основная программа. Вообще процедуру нужно воспринимать как программу в миниатюре. В свою очередь, основная программа может быть легко переделана в процедуру с заменой слова PROGRAM на PROCEDURE. Если процедура объявлена, ее можно использовать в последующих частях программы, просто записывая ее имя, за которым, если необходимо, следует список параметров. Вызов процедуры для основной программы становится новым оператором. Обращение к процедуре активизирует эту процедуру, т. е. приводит к выполнению группу операторов, содержащихся в ее теле. После этого управление переходит к оператору, следующему за вызовом процедуры.

1.1.3. Процедуры без параметров

Заголовок процедуры без параметров выглядит как:

PROCTDURE <Имя процедуры>;

Вызываются такие процедуры путем написания в основной программе имени этой процедуры. В виде процедуры без параметров оформляются такие подзадачи, у которых нет входных и выходных данных, или же эти данные удобнее передавать с помощью операторов присваивания: READ и WRITE.

Рассмотрим несколько примеров, в которых представлены эти варианты.

П р и м е р 1. Нарисовать три вертикальных квадрата 3 × 3 с помощью символа «*».

Очевидно, что в этой программе надо выделить рисование квадрата в виде процедуры без параметров, а затем трижды вызвать ее в основной программе.

```
program RISUNOK;
procedure KVADRAT;
begin
writeln ('***');
writeln ('* *');
writeln ('***');
end;
```

```
begin
clrscr; KVADRAT;
writeln; KVADRAT;
writeln; KVADRAT;
end.
```

П р и м е р 2 . Вычислить площадь четырехугольника ABCD.

Зная длины сторон четырехугольника и длину одной из его диагоналей, например BD, можно по формуле Герона найти площади двух вспомогательных треугольников и сложить их. Отсюда следует, что в программе надо выделить процедуру вычисления площади треугольника.

```
program PLOCHAD_1;
var AB, BC, CD, AD, BD, S1, S, a, b, c, p:real;
procedure GERON_1;
begin
p := (a + b + c)/2;
S := sqrt (p*(p – a)*(p – b)*(p – c));
end;
begin {*ОСНОВНАЯ ПРОГРАММА*}
read (AB, BC, CD, AD, AC);
a := AB; b := AD; c := BD; GERON_1; S1:= S;
a := BC; b := CD; c := BD; GERON_1; S1:= S1 + S;
write (S1);
end.
```

Примечание. В данной программе все вычисления проходят с помощью переменных, объявленных в разделе VAR основной программы.

1.1.4. Процедуры с параметрами-значениями

Как было сказано ранее, процедуры с параметрами-значениями требуют входных данных. Где они записываются и как задаются? На этот вопрос может ответить общая форма заголовка процедуры этой процедуры:

PROCEDURE <Имя процедуры> (<Параметры-аргументы>: тип);

Здесь под параметром понимают имя переменной, которая является «входной» для процедуры (формальный параметр-аргумент). Этот параметр с синтаксической точки зрения является параметром-значением, при его описании в заголовке процедуры не требуется писать слово VAR. Параметры-значения при вызове процедуры принимают из основной программы свои конкретные значения. Заме-

тим также, что в самой процедуре значения параметров-значений не меняются в ходе работы процедуры.

При обращении к процедуре с параметрами-значениями в основной программе фактическими параметрами могут служить как имена переменных (которые описаны и определены выше), так и конкретные значения (константы) и выражения. При обращении необходимо следить за соответствием списка параметров при обращении и описании. Кроме того, следует строго соблюдать соответствие типов параметров.

Рассмотрим работу процедур такого типа на примерах.

П р и м е р 1. Нарисовать квадрат с произвольной длиной стороны в левом верхнем углу (длина стороны задается с клавиатуры).

В этой программе также надо оформить рисование квадрата в виде процедуры, но уже с входным параметром-значением – длиной стороны квадрата.

```
program RISUNOK_2;
var I: integer;
procedure KVADRAT (N: integer);
var J, K: integer;
begin
for J := 1 to N do write ('*'); writeln;
for J := 1 to N – 2 do
begin
write ('*'); for K := 1 to N – 2 do write (' ');
writeln ('*');
end;
for J := 1 to N do write ('*');
end;
begin { Основная программа }
write ('Введите длину стороны – ');
readln (I); clrscr; KVADRAT (I);
end.
```

П р и м е р 2. Вычислить площадь четырехугольника с применением процедуры с параметрами-значениями.

```
program PLOCHAD_2;
var AB, BC, CD, AD, AC, S1, S: real;
procedure GERON_2 (a, b, c: real);
var P: real;
begin
P := (a + b + c)/2; S := sqrt (P*(P – a)*(P – b)*(P – c));
end;
```

```
begin {*ОСНОВНАЯ ПРОГРАММА*}
read (AB, BC, CD, AD, AC); GERON_2(AB, BC, AC); S1:= S;
GERON_2 (AD, AC, CD); write ('S = ', S1 + S)
end.
```

В данной программе определена процедура GERON_2 с тремя параметрами-значениями и локальной переменной P. Значение же площади треугольника помещается в глобальную переменную S. При вызове этой процедуры формальные параметры a, b, c замещаются на фактические параметры AB, BC, AC при первом обращении и на AD, AC, CD – при втором.

Заметим также, что здесь фактические параметры представлены переменными, которые получают свое значение с помощью процедуры READ. Однако если известны длины сторон треугольника, например, 6, 7, 4, то можно вычислить площадь этого треугольника, вызвав процедуру GERON_2(6, 7, 4), и получить ответ в переменной S.

1.1.5. Процедуры с параметрами-переменными

В отличие от процедур с параметрами-значениями, процедуры с параметрами-переменными не имеют входных параметров, т. е. из основной программы не передаются значения переменных в процедуру, за исключением глобальных переменных. Отличие в описании и обращении к процедурам с параметрами-переменными заключается в специфическом написании заголовка процедуры. В остальном все процедуры схожи. Общий вид заголовка процедуры с параметрами-переменными выглядит так:

PROCEDURE <Имя процедуры> (VAR<Параметры-переменные>: тип);

При детальном ознакомлении с синтаксической диаграммой видно, что параметрам-переменным должно предшествовать служебное слово VAR, причем оно пишется столько раз, сколько различных типов представлено в выходных данных, например:

PROCEDURE PRIMER (VAR a, b, c : INTEGER; VAR m : CHAR; VAR i, j : REAL).

При обращении к процедурам с параметрами-переменными фактическими параметрами должны являться имена переменных, которые описаны в основной программе.

П р и м е р . Вычисление площади четырехугольника.

```
program PLOCHAD_3;
var AB, BC, CD, AD, AC, S1, S2, a, b, c: real;
procedure GERON_3 (var S: real);
```

```
var P: real;
begin
P := (a + b + c)/2; S := sqrt (P*(P – a)*(P – b)*(P – c));
end;
begin { Основная программа }
read (AB, BC, CD, AD, AC);
a := AB; b := BC; c := AC; GERON_3 (S1);
a := AD; b := AC; c := CD; GERON_3 (S2);
write ('S = ', S1 + S2)
end.
```

1.1.6. Комбинированные процедуры

Комбинированные процедуры включают в себя входные и выходные данные. В заголовке этой процедуры, как и ранее, выходные параметры предваряются словом VAR, входные параметры – без него. Порядок следования параметров может быть произвольным, например:

```
PROCEDURE PRIMER (VAR a, b, c: INTEGER;
m: CHAR; VAR i, j: REAL);
```

Здесь a, b, c, i, j – параметры-результаты (переменные); m – параметр-аргумент (значение).

В качестве иллюстрации комбинированных процедур рассмотрим последний вариант вычисления площади четырехугольника:

```
program PLOCHAD_4;
var AB, BC, CD, AD, AC, S1, S2: real;
procedure GERON_4 (a, b, c : real; var S : real);
var P : real;
begin
P := (a + b + c)/2;
S := sqrt (P*(P – a)*(P – b)*(P – c));
end;
begin {*ОСНОВНАЯ ПРОГРАММА*}
read (AB, BC, CD, AD, AC);
GERON_4 (AB, BC, AC, S1);
GERON_4 (AD, AC, CD, S2);
write ('S = ', S1 + S2)
end.
```

Примечание. Для более полного усвоения введенных ранее терминов перечислим на базе последнего примера все виды параметров и переменных:

1) глобальные переменные AB, BC, CD, AD, AC, S1, S2;
2) локальные переменные a, b, c, S, P;
3) формальные параметры a, b, c, S:
– параметры-значения (аргументы) a, b, c;
– параметр-переменная (результат) S;
4) фактические параметры AB, BC, CD, AD, AC, S1, S2:
– параметры-значения (аргументы) AB, BC, CD, AD, AC;
– параметры-переменные (результаты) S1, S2.

Заметим также, что термины «параметр-значение» и «аргумент», как и «параметр-переменная» и «результат», не всегда идентичны. Дело в том, что характеристика «значение (переменная)» отражает синтаксическую сущность параметра, а «аргумент (результат)» – его семантику (функциональную роль в процедуре). Иногда один и тот же параметр может быть аргументом и результатом одновременно, но описан в процедуре в виде параметра-переменной.

Попытка же описать выходной параметр в виде параметра-значения (без слова VAR в заголовке процедуры) приведет к тому, что результат работы процедуры не будет возвращен в основную программу. Это происходит потому, что характер «поведения» параметров-значений и параметров-переменных в процессе работы процедуры различен. Разница эта состоит в том, что преобразования, которые претерпевают формальные параметры-значения в процедуре, не вызывают изменения соответствующих им фактических параметров, в то время как изменения параметров-переменных могут изменять значения соответствующих фактических параметров.

Причиной этого феномена является неодинаковое распределение памяти под хранение параметров процедуры. Формальному параметру-значению отводится некоторая область (ячейка) памяти, куда заносится значение соответствующего фактического параметра, вычисленного на момент обращения к процедуре. На этом связь между ними обрывается. Действительно, если фактическим параметром является константа или выражение, то как изменения в формальном параметре-значении (а это есть всегда переменная) могут повлиять, например, на выражение?

Фактическим же параметром, соответствующим формальному параметру-переменной, является всегда переменная. На время выполнения процедуры эти параметры отождествляются, им соответствует одна и та же область памяти. Вполне понятно, что в этой ситуации изменения формального параметра влекут адекватные изменения фактического параметра, и после завершения процедуры его значение может отличаться от его первоначального значения.

Именно поэтому, объявив в процедуре параметр-результат как параметр-значение, этот результат так и останется в формальном параметре-переменной без его передачи в соответствующий фактиче-

ский параметр (т. е. нет возврата полученного процедурой значения) в основную программу.

1.1.7. Задание для самостоятельного решения. Процедуры

1. Заданы два вектора $\mathbf{x = (x_1, x_2, x_3, x_4)}$, $\mathbf{y = (y_1, y_2, y_3, y_4)}$. Определить угол **a** между векторами × и **y** по формуле: $\alpha = \arccos\frac{(x,y)}{\sqrt{(x,x)(y,y)}}$ Вычисление скалярного произведения оформить в виде процедуры.

2. Четыре точки заданы своими координатами $\mathbf{X(x_1, x_2)}$, $\mathbf{Y(y_1, y_2)}$, $\mathbf{Z(z_1, z_2)}$, $\mathbf{P(p_1, p_2)}$. Выяснить, какие из них находятся на максимальном расстоянии друг от друга и вывести на печать значение этого расстояния. Вычисление расстояния между двумя точками оформить в виде процедуры.

3. Четыре точки заданы своими координатами $\mathbf{X(x_1, x_2, x_3)}$, $\mathbf{Y(y_1, y_2, y_3)}$, $\mathbf{Z(z_1, z_2, z_3)}$, $\mathbf{T(t_1,t_2, t_3)}$. Выяснить, какие из них находятся на минимальном расстоянии друг от друга и вывести на печать значение этого расстояния. Вычисление расстояния между двумя точками оформить в виде процедуры.

4. Три точки заданы своими координатами $\mathbf{X(x_1, x_2)}$, $\mathbf{Y(y_1, y_2)}$ и $\mathbf{Z(z_1, z_2)}$. Найти и напечатать координаты точки, для которой угол между осью абсцисс и лучом, соединяющим начало координат с точкой, минимальный. Вычисления оформить в виде процедуры.

5. Задана окружность $\mathbf{(x-a)^2 + (y-b)^2 = R^2}$ и точки $\mathbf{P(p_1, p_2)}$, $\mathbf{F(f_1, f_1)}$, $\mathbf{L(l_1,l_2)}$. Выяснить и напечатать, сколько точек лежит внутри окружности. Проверку, лежит ли точка внутри окружности, оформить в виде процедуры.

6. Даны отрезки **a**, **b**, **c** и **d**. Для каждой тройки этих отрезков, из которой можно построить треугольник, вывести на экран площадь данного треугольника. Проверку существования треугольника оформить в виде процедуры.

7. Даны длины сторон треугольника **a**, **b**, **c**. Найти медианы треугольника, сторонами которого являются медианы исходного треугольника. Для вычисления медианы проведенной к стороне **a**, использовать формулу $0{,}5\sqrt{2b^2 + 2c^2 - a^2}$ Вычисление медианы оформить в виде процедуры.

1.2. Функции пользователя. Рекурсивные функции

Функция как объект языка Паскаль является другой версией реализации технологии построения программ с использованием структуры группирования. Можно также сказать, что функция есть частный вид

определенного типа процедур, а именно процедур с одним параметром-переменной.

1.2.1. Определение функции

Функция отличается от процедуры только тем, что всегда возвращает в точку вызова одно скалярное значение. При этом функция, как и процедура, может содержать параметры-значения или быть без них.

Общая форма записи заголовка функции FUNCTION имя (список параметров: тип): тип; или FUNCTION имя: тип;

Тип результата есть тип значения функции. Список параметров такой же, что и для процедуры, только здесь все параметры являются аргументами. Имя переменной, которая хранит значение функции, совпадает с именем функции.

Итак, заголовок функции отличается от заголовка процедуры не только сменой слова PROCEDURE на FUNCTION, но и удалением из списка параметров параметра-результата с присвоением его типа имени функции.

Другой особенностью описания функции является наличие в нем хотя бы одного оператора присваивания, в левой части которого стоит имя определяемой функции, а в правой – выражение для вычисления результата функции. Очевидно, что тип этого выражения должен совпадать с указанным в заголовке типом функции.

Вызов функции также отличается от вызова процедуры. Если вызов процедуры осуществляется с помощью специального оператора вызова (оператора процедуры), то функция вызывается только внутри некоторого выражения. Для того, чтобы осуществить обращение к функции, необходимо использовать ее имя со списком фактических параметров в каком-либо выражении, тип которого совпадает с типом значения функции. Само же выражение, внутри которого вызывается функция, может быть правой частью оператора присваивания, частью логического выражения и пр.

1.2.2. Функции пользователя

Известно, что Паскаль имеет набор стандартных функций. Однако этот набор ограничен. Пользователь может по желанию расширить список функций, создав свои функции – функции пользователя. Так, например, в Паскале есть SQR (X) = X^2, а вот функции F (X) = X^n, где n принадлежит множеству целых чисел Z, нет. Используя определенное ранее понятие функции, можно создать для этого универсальную функцию, которая давала бы степени произвольного вещественного числа с любым целым показателем.

Определим вещественную функцию POWER, которая должна иметь два параметра-аргумента – для показателя и для основания степени:

```
function POWER (FACTOR: real; EXP: integer): real;
var COUNT: integer; TFACTOR: real;
begin
if EXP = 0 then POWER := 1
else begin
TFACTOR := FACTOR;
for COUNT := 2 to ABS (EXP) do
TFACTOR := TFACTOR*FACTOR;
if EXP < 0 then POWER := 1/TFACTOR
else POWER := TFACTOR
end
end;
```

Теперь можно эту функцию вызывать следующим образом:

а) F := POWER(5.25, 3);
б) WRITELN ("D = ", POWER (5.25, -2):5:2);
в) IF X > 2*POWER (6.2, 3) THEN WRITE ('ДА');
г) A := POWER (X, 2) + POWER (X, 3) + POWER (X, 4).

1.2.3. Рекурсивные функции

К функциям можно обращаться тремя способами: из тела основной программы, из тела другой функции, из тела самой функции, т. е. функция может вызывать саму себя. Функции называются *рекурсивными*, если в описании функции происходит вызов самой себя, а процесс обращения – *рекурсией*. Продемонстрируем использование рекурсии на примере вычисления значения факториала произвольного натурального числа N.

В математике известно рекурсивное определение факториала:

$n! = 1$ при $n = 0$;

$n! = (n – 1)! \cdot n$ при $n > 0$.

Это рекурсивное определение можно реализовать с помощью соответствующей рекурсивной функции:

```
function FACTORIAL (VALUE: integer): integer;
begin
iF VALUE = 0 then FACTORIAL := 1
else FACTORIAL := VALUE*FACTORIAL (VALUE – 1)
end;
```

Теперь можно обращаться к этой функции в теле основной программы, как показано в следующем примере:

```
program FINDFACTORIAL;
var N: integer;
begin
writeln ('Введите число');
readln (N);
if N < 0 then writeln ('Нет факториала')
else writeln ('Факториал ', N, ' равен ', FACTORIAL (N))
end.
```

Мы видим, что характерной особенностью построенной функции является наличие в ее теле оператора присваивания:

FACTORIAL := VALUE*FACTORIAL (VALUE – 1),

где происходит вызов определяемой функции. Здесь идентификатор FACTORIAL в левой части оператора обозначает имя переменной для хранения значения функции, а в правой – имя вызываемой функции.

Важным моментом при составлении любой рекурсивной функции является организация выхода из рекурсии. В некоторых простых случаях должно существовать не рекурсивное решение. Рекурсивный процесс должен шаг за шагом так упрощать задачу, чтобы, в конце концов, для нее появилось не рекурсивное решение. В этих функциях должны проверяться значения аргумента для принятия решения о завершении. В нашем случае условием завершения рекурсии является VALUE = 0.

При описании рекурсивных функций необходимо хорошо представлять процесс вычислений. Всякая рекурсия состоит из двух этапов: углубления (погружения) внутрь рекурсии и выхода из нее. На первом этапе никаких вычислений не производится, а идет только настройка рабочей формулы на конкретные операнды. На втором этапе происходит процесс вычислений по настроенным формулам.

В данном примере покажем, что часто рекурсивные функции строятся гораздо проще, чем «обычные», хотя вполне понятно, что не всякая функция может быть переделана на рекурсивную. Сделаем это на примере уже построенной ранее функции POWER.

Данная функция явно носит рекурсивный характер, исходя из ее определения:

$X^n = 1$, если $n = 0$;
$X^n = X^{n-1} * X$, если $n > 1$.

Ниже следует рекурсивная функция вычисления значения степени:

```
function POWER (FACTOR: real; EXPONENT: integer): REAL;
begin
if EXPONENT < 0
then POWER := 1/POWER (FACTOR, abs (EXPONENT))
else
if EXPONENT > 0
then POWER := FACTOR*POWER (FACTOR, EXPONENT – 1)
else POWER := 1
end;
```

Замечание. Помимо рекурсивных функций, в языке Паскаль по тому же принципу можно определять и рекурсивные процедуры. Пример как рекурсивная функция может быть переделана в рекурсивную процедуру на примере вычисления факториала:

```
procedure FACTORIAL (VALUE: integer; var F: integer);
begin
iF VALUE = 0 then F := 1
else begin FACTORIAL (VALUE – 1, F);
F := F*VALUE
end;
end;
```

Здесь уже, в отличие от функции FACTORIAL, для вычисления N! необходимо вызвать эту процедуру с помощью оператора процедуры FACTORIAL (N, FN), где FN – переменная для возвращения из процедуры значения N!.

Примеры реализации программ

Пример 1. Вычислить значения функции $f(x)=2\cos x+3$, при $x=\{1; 4; 7{,}5; 20\}$. Вывести результаты в два столбца: в первом – значения x, во втором – значения $f(x)$. Вычисления провести двумя способами: с помощью функции и процедуры.

Решение. Аргумент и результат функции – действительные числа, поэтому используем тип Real. В теле функции будет только оператор присваивания – для вычисления значения выражения. Процедура отличается строкой заголовка, – для передачи в основную программу результатов вычислений добавим параметр-переменную *fx*. Чтобы вывести результаты в виде таблицы, используем форматный вывод.

```
program proc_1;
function f (x: Real):Real;
begin
 f:=2*cos(x)+3
end;
procedure proc_f (x: Real; var fx: real);
begin
 fx:=2*cos(x)+3
end;
 var x, fx: real;
begin
 Writeln('с использованием процедуры:');
 Writeln(' ×  f(x)');
 x:=1; proc_f (x,fx); Writeln (x:6:2, fx:6:2);
 x:=4; proc_f (x,fx); Writeln (x:6:2, fx:6:2);
 x:=7.5; proc_f (x,fx); Writeln (x:6:2, fx:6:2);
 x:=20; proc_f (x,fx); Writeln (x:6:2, fx:6:2);
 Readln;
 Writeln('с использованием функции:');
 Writeln(' × f(x)');
 Writeln(1:6, f (1):6:2);
 Writeln(4:6, f (4):6:2);
 Writeln(7.5:6:2, f (7.5):6:2);
 Writeln(20:6, f (20):6:2);
  Readln;
end.
```

Пример 2. Создать рекурсивную функцию поиска i-го члена последовательности, заданной рекуррентной формулой A_1=const1, A_2=const2, $A_i=3A_{i-2}-A_{i-1}$. Вывести через пробел значения рекурсивной функции при значениях аргумента от 1 до 10 вномерамительно.

Решение. По условию задачи аргумент может принимать только целые значения, поэтому функция имеет параметр-значение типа Integer. Выход из рекурсии в данном случае осуществляется при двух значениях аргумента (при *i*=1, *i*=2), поэтому в рекурсивной функции необходимы два вложенных условных оператора или же оператор выбора *case*. В приведенном примере использованы операторы *if*, но попробуйте самостоятельно записать решение с помощью оператора выбора. В основной программе значения аргумента – целые последовательные числа, поэтому следует воспользоваться оператором цикла с параметром *for*.

```
program proc_2;
function A (i: Integer): Integer;
begin
```

```
 if i=1 then A:=1 else
 if i=2 then A:=3 else A:=3*A(i-2)-A(i-1)
end;
 var i: Integer;
begin
 for i:=1 to 10 do Write (A(i),' ');
 Readln
end.
```

1.2.4 Задания для самостоятельного выполнения. Функции

1. Даны действительные числа **a**, **b**, **c**. Получить:

$$\frac{\max(a, a+b) + \max(a, b+c)}{1 + \max(a+bc, b, 15)}$$

2. Даны действительные числа **a**, **b**. Получить **u = min(a, b-a)**, **y = min(ab, a+b)**, **k = min(u+v², 3.14)**.

3. Даны натуральные числа **a**, **b**, **c**. Определить функцию **bin(x)**, переводящую число **x** из десятичной системы счисления в двоичную. Найти **bin(a + b)**, **bin(ab + c)**.

4. Даны действительные числа **s**, **t**. Получить: **g(1.2, s)+g(t, s)-g(2s – 1.5t)**, **|g(ln(s, t+1))-g(t, s)|**, где

$$g(a,b) = \frac{a^2 + b^2}{a^2 + 2ab + 3b^2 + 4}$$

5. Даны действительные числа **x**, **y**. Получить: **f(x, -2y, 1.17)+f(2.2, x, x-y)**, **tg(f(x+y, xy, y-x)+f(3.1, 1.4, y-sinx))**, где

$$f(a,b,c) = \frac{2a - b - \sin c}{5 + |c|}.$$

6. Даны натуральные числа **a**, **b**, **c**. Найти **НОД(a, b, c)**, используя формулу: **НОД(a, b, c) = НОД((a, b), c)**.

7. Даны неотрицательные целые числа **a**, **b**. Найти **F(a, b)**, где $F(m,n) = \frac{n! - m!}{(n+m)!}$. (Определить вспомогательную функцию, вычисляющую факториал).

8. Даны два натуральных числа **a**, **b**. Найти разность и произведение суммы цифр этих чисел. Вычисление суммы цифр числа оформить в виде функции.

9. Даны два натуральных числа **a**, **b**. Вычислить $\frac{a!! - ab}{a!! + ab}$. Функция **x!!** Определяется следующим образом: **x!! = 1*3*5*...*x**, если **x** нечетно, **x!! = 2*4*6*...*x**, если **x** четно.

10. Даны действительные числа a_0, a_1, a_2, a_3. Получить для **x = 1, 3, 4** значения **p(x+1) – p(x)**, где $p(y) = a_3y^3 + a_2y^2 + a_1y + a_0$.

11. Даны действительные числа **a**, **b**, **c**. Получить $\frac{\min(a,a+b)+\min(a,b+c)}{1+\min(a+bc,b)}$.

12. Даны действительные числа **a**, **b**. Получить **r = max (a, b + a)**, **d = max (ab, a + b)**, **s = max (r + d^2, 3.14)**.

13. Даны действительные числа **a**, **b**. Получить:

$$g(1.2,a)+g(b,a)-g(2a-1,5b);$$

$$\left|g(\ln(ab+1))-g(b,a)\right|, \quad где$$

$$g(p,r)=\frac{p^2+r^2}{p^2+2pr+3r^3+4}.$$

1.3. Задачи для самостоятельного решения. Общие задачи по теме процедуры и функции

1. Составить программу для решения задачи с применением функции и процедуры пользователя.

a) В правильном треугольнике проведена средняя линия. Найти площадь образовавшейся трапеции, дважды используя функцию вычисления площади правильного треугольника по формуле:

$$s=\frac{a^2\sqrt{3}}{4}$$

b) Для правильного треугольника со стороной *a* построены вписанная и описанная окружности. Найти площадь образовавшегося кольца, используя функцию вычисления площади круга $S=\pi R^2$. Для нахождения радиусов окружностей воспользуйтесь формулами:

$$R=\frac{a\sqrt{3}}{3}, r=\frac{a\sqrt{3}}{6}.$$

c) Тариф предусматривает оплату телефонных разговоров следующим образом: при продолжительности разговора меньше *P* минут стоимость одной минуты составляет S_1 копеек, в противном случае – S_2 коп/мин (S_1, S_2, *P* – константы). Используя функцию вычисления стоимости одного разговора, найти суммарную стоимость трех звонков известной продолжительности.

d) На товар дважды была сделана скидка – на p_1, а затем на p_2 процентов. Первоначальная стоимость товара составляла *S* рублей. Используя функцию вычисления стоимости товара с учетом скидки на *P* процентов, найти стоимость товара после двойной скидки.

$$C=\frac{100-P}{100}\cdot S$$

2. Найти член последовательности, заданной формулой: $D_i=7+D_{i-1}$ при i>1, где D_1 определяется пользователем.

3. Найти член последовательности, заданной формулой: $A_i=A_{i-1}-A_{i-2}$ при i>2. Значения первого и второго членов последовательности вводятся пользователем.

4. Найти член последовательности, заданной следующим образом: $y_1=0$; $y_2=10$; $y_n=2\cdot y_{n-1}-y_{n-2}$, где n>2.

5. Найти член последовательности, заданной формулой $B_i=4\cdot B_{i-1}$, при i>1. Значения первого члена последовательности вводится пользователем.

6. Для чисел *a, b, c* найти значение выражения *min(a,ab)+min(a,ac)+1* с использованием функции вычисления минимального из двух чисел.

7. Дан квадрат со стороной *a*, диагональ этого квадрата является стороной второго квадрата, диагональ второго квадрата – стороной третьего. Найти длину стороны третьего квадрата, используя функцию вычисления длины диагонали квадрата по его стороне: $d = a\sqrt{2}$

8. Вычислить $C_n^k = \frac{n!}{k!\,(n-k)!}$. Вычисление факториала числа оформите в виде подпрограммы.

9. Построить таблицу значений функции $z = \frac{sh(x+y) - sh(x-2y)}{sh^2(x-y)}$, где × меняется от **1** до **2** с шагом **0,2**; **y** меняется от **2** до **3** с шагом **0,1**.

Вычисление гиперболического синуса $sh\ k = \frac{e^k - e^{-k}}{2}$ оформить в виде подпрограммы.

10. Построить таблицу значений функции $z = \frac{ch(x-y) - ch(x^2-y)}{ch^2(x) + ch(x+y)}$, где × меняется от **3** до **4** с шагом **0,1**; **y** меняется от **2** до **3** с шагом **0,2**.

Вычисление гиперболического косинуса $ch\ k = \frac{e^k + e^{-k}}{2}$ оформить в виде подпрограммы.

11. Найдите **НОД(a, b, c, d) = НОД(НОД(a, b), НОД(c, d))**. Нахождение **НОД** (наибольшего общего делителя) двух чисел оформите в виде подпрограммы.

12. Найдите **НОК(a, b, c, d) = НОК(НОК(a, b), НОК(c, d))**. Нахождение **НОК** (наименьшего общего кратного) двух чисел оформите в виде подпрограммы.

13. Найти все коэффициенты разложения **(x+y)**^{n}, т. Е. C_n^k, где **k= 0, ..., n**. Вычисление C_n^k оформите в виде подпрограммы.

14. Даны длины отрезков **a,b,c,d,e**. Выяснить, можно ли построить треугольники со сторонами **{a,b,c}**, **{b,c,d}**, **{c,d,e}**? Если да, то найдите площадь соответствующего треугольника. Проверку на возможность составления треугольника и вычисление площади оформите в виде подпрограммы.

15. Вычислить $A_n^k = \frac{n!}{(n-k)!}$ и $P_n = n!$. Нахождение факториала числа оформите в виде подпрограммы.

16. Проверить, будут ли числа **a, b, c, d** попарно взаимно просты. Числа называются взаимно простыми, если их наибольший общий делитель **(НОД)** равен **1**. Нахождение **НОД** двух чисел оформить в виде подпрограммы.

17. Треугольник задан координатами своих вершин. Составить программу вычисления его площади.

18. Вычислить площадь правильного шестиугольника по заданной стороне, используя подпрограмму нахождения площади треугольника.

19. Написать программу нахождения суммы факториалов всех нечетных чисел от 1 до 9.

20. Даны две дроби $\frac{A}{B}$ и $\frac{C}{D}$ (A, B, C, D – натуральные числа). Составьте программу:

а) деления дроби на дробь;

б) умножения дроби на дробь;

в) сложения этих дробей.

Ответ должен быть несократимой дробью.

21. Дано простое число. Составить функцию, которая буден находить следующее за ним простое число.

22. Составить функцию для нахождения наименьшего нечетного натурального делителя k ($k \neq 1$) любого заданного натурального числа n.

23. Составить программу, определяющую, в каком из данных двух натуральных чисел больше цифр.

24. Два натуральных числа называются «дружественными», если каждое из них равно сумме всех делителей (кроме его самого) другого (например, числа 220 и 284). Найти все пары «дружеских чисел», которые не больше данного числа N.

25. Два простых числа называются «близнецами», если они отличаются друг от друга на 2 (например, 41 и 43). Напечатать все пары «близнецов» из отрезка [n, 2n], где n – заданное натуральное число больше 2.

26. Написать программу вычисления суммы

$\frac{p}{q} = 1 - \frac{1}{2} + \frac{1}{3} - \ldots + \frac{(-1)^{n+1}}{n}$ для заданного числа n. Дробь $\frac{p}{q}$ должна быть несократимой (p, q – натуральные числа).

27. Написать программу вычисления суммы $1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}$ для заданного числа n. Результат представить в виде несократимой дроби $\frac{p}{q}$ (p, q – натуральные).

28. Натуральное число в записи которого n цифр, называется числом Амстронга, если сумма его цифр, возведенная в степень n, равна самому числу. Найти все эти числа от1 до k.

29. Написать программу, которая находит и выводит на печать все четырехзначные числа вида $\overline{abcd}$, для которых выполняются условия: a, b, c, d – разные цифры; $\overline{ab} - \overline{cd} = a + b + c + d$.

30. Найти все натуральные числа, не превосходящие n, двоичная запись которых представляет собой палиндром, т. е. читается одинаково слева направо и справа налево.

31. Найти все натуральные n-значные числа, цифры в которых образуют строго возрастающую последовательность (1234, 5789, 5689 и т. д.)

32. Найти все натуральные числа, не превосходящие заданного n, которые делятся на каждую из своих цифр.

33. Составить программу нахождения чисел из интервала [M, N], имеющих наибольшее количество делителей.

34. Для последовательности $a_1=1$, $a_{n+1} = a_n + \frac{1}{1+a_n}$ составить программу печати k-ого члена в виде обыкновенной несократимой дроби. Например, $a_2 = \frac{3}{2}, a_3 = \frac{19}{20}$

35. Дано натуральное число n. Выяснить, можно ли его представить в виде произведения трех последовательных натуральных чисел.

36. Напишите программу, состоящую из трех процедур и основной программы. Первая процедура организует ввод двух целых чисел X и Y, вторая проверяет их сумму, третья выводи результат. Используйте эти процедуры в основной программе. Используйте X и Y как глобальные переменные.

37. Напишите программу вычисления площади поверхности и длины экватора на основе известного радиуса планет солнечной

системы. Форму планет будем считать шаром. Вычисление площади поверхности и длины экватора оформите отдельными функциями.

38. Напишите программу вычисления суммы: 1! + 2! + 3! + … + n!, используя функцию вычисления факториала числа k.

39. Напишите программу для вычисления числа сочетаний из N по M. Число сочетаний определяется по формуле N!/(M!*(N-M)!, где N – количество элементов перебора. Используйте подпрограмму вычисления факториала.

40. Напишите программу для определения НОД трех натуральных чисел.

41. Даны действительные числа s,t. Составить программу вычисления выражения f(t, -2s, 1.17) + f(2.2, t, s-t), где f(a,b,c) = (2a – b – sin(c)) / (5 + |c|).

42. Даны натуральные m и n (m<n). Составить программу, сокращающую дробь m/n.

43. Напишите программу вычисления суммы квадратов простых чисел, лежащих в интервале (M,N).

44. Напишите программу подсчета числа четных цифр, используемых в записи N-значного числа M.

45. Составьте программу вычисления суммы трехзначных чисел, в десятичной записи которых нет четных цифр.

46. Составьте программу вывода на экран всех натуральных чисел, не превосходящих N и делящихся на каждую из своих цифр.

47. Составьте программу нахождения наименьшего натурального N-значного числа X (X>=10), равного утроенному произведению своих цифр.

48. Составьте программу подсчета числа всех натуральных чисел, меньших M, квадрат суммы цифр которых равен X.

49. Треугольник задан координатами своих вершин. Составить программу для вычисления его площади.

50. Составить программу для нахождения наибольшего общего делителя и наименьшего общего кратного двух натуральных чисел

$$(НОК(A,B) = \frac{A \cdot B}{НОД(A,B)})$$

51. Составить программу для нахождения наибольшего общего делителя четырех натуральных чисел.

52. Составить программу для нахождения наименьшего общего кратного трех натуральных чисел.

53. Написать программу для нахождения суммы большего и меньшего из трех чисел.

54. Вычислить площадь правильного шестиугольника со стороной a, используя подпрограмму вычисления площади треугольника.

55. Написать программу для вычисления суммы факториалов всех нечетных чисел от 1 до 9

56. Даны числа X, Y, Z, T – длины сторон четырехугольника. Вычислить его площадь, если угол между сторонами длиной X и Y – прямой.

57. Составить программу для вычисления суммы факториалов всех четных чисел от m до n.

58. Дано простое число. Составить функцию, которая будет находить следующее за ним простое число.

59. Заменить данное натуральное число на число, которое получается из исходного записью его цифр в обратном порядке (например, дано число 156, нужно получить 651).

60. Найти все натуральные n-значные числа, цифры в которых образуют строго возрастающую последовательность (например, 1234, 5789).

61. Найти все натуральные числа, не превосходящие заданного n, которые делятся на каждую из своих цифр.

62. Дано натуральное число n. Выяснить, можно ли представить его в виде произведения трех последовательных натуральных чисел.

63. Имеется часть катушки с автобусными билетами. Номер билета шестизначный. Составить программу, определяющую количество счастливых билетов на катушке, если меньший номер билета – N, больший – M (билет является счастливым, если сумма первых трех его цифр равна сумме последних трех).

64. Написать программу, определяющую сумму n-значных чисел, содержащих только нечетные цифры. Определить также, сколько четных цифр в найденной сумме.

65. Из заданного числа вычли сумму его цифр. Из результата вновь вычли сумму его цифр и т. д. Сколько таких действий надо произвести, чтобы получился нуль?

66. Составить программу для разложения данного натурального числа на простые множители. Например, 200 = 23 • 52.

67. Дано натуральное число n. Найти все числа Мерсена меньшие n. (Простое число называется числом Мерсена, если оно может быть представлено в виде 2p – 1, где p – тоже простое число. Например, 31 = 25 – 1 – число Мерсена.)

68. Дано четное число n> 2. Проверить для него гипотезу Гольдбаха: каждое четное n представляется в виде суммы двух простых чисел.